The Randomized z-Buffer Algorithm

Interactive Rendering of Highly Complex Scenes

SIGGRAPH 2001
EXPLORE INTERACTION AND DIGITAL IMAGES

Michael Wand   Matthias Fischer   Ingmar Peter
Friedhelm Meyer auf der Heide   Wolfgang Straßer

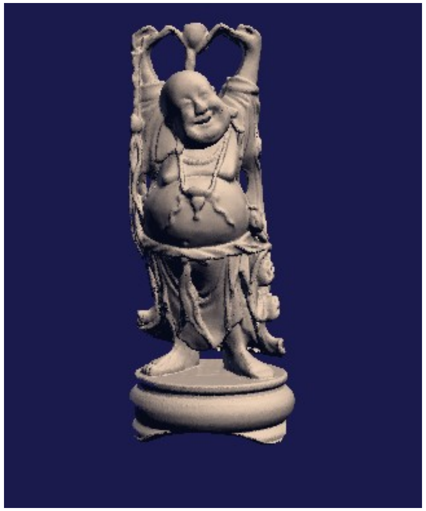
WS GRIS

WSI / GRIS
University of Tübingen

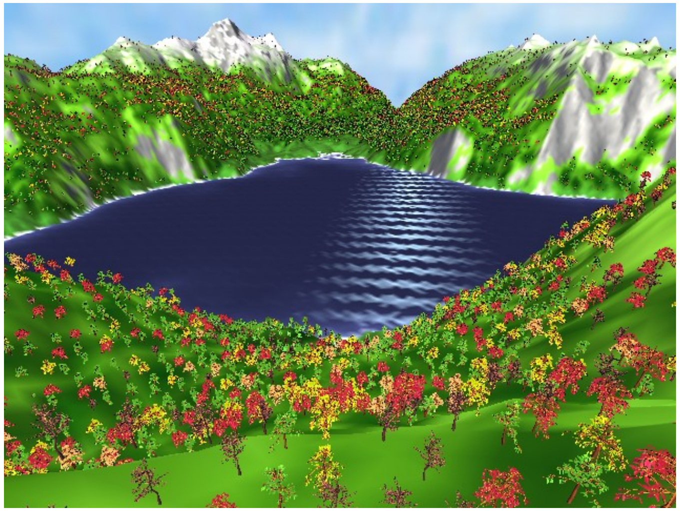
Heinz Nixdorf Institute
University of Paderborn

Introduction

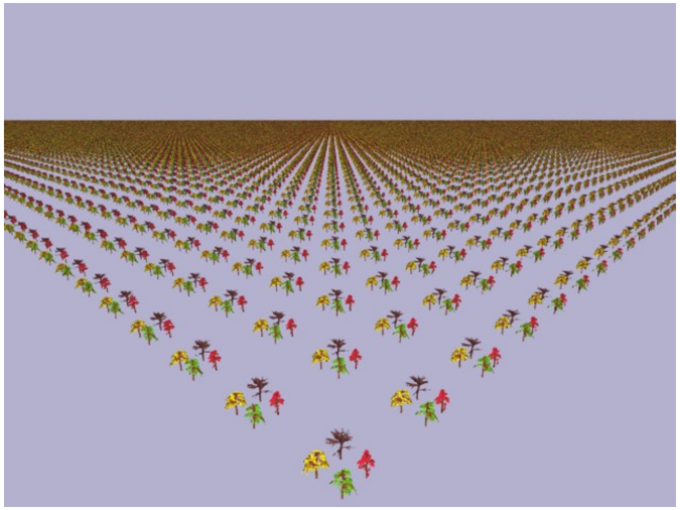
Scene Complexity



10^6 triangles



10^8 triangles



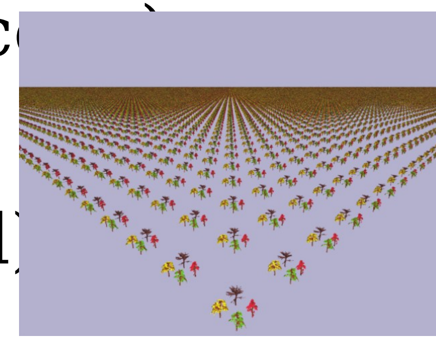
10^{14} triangles

Highly detailed scenes:

- Visualization, Games, CAD, ...
- Interactive walkthrough, editing
- Efficient rendering needed

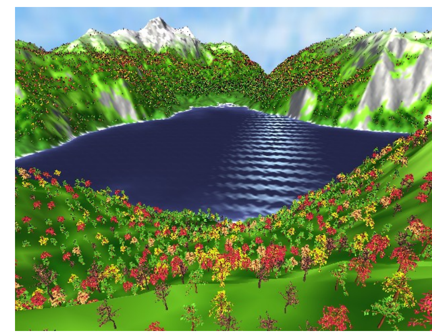
Output-Sensitive Rendering

Complexity parameters: (triangle sc



- Number of triangles: n
- Projected area (visible + occluded): a

Z-Buffer-Algorithm:

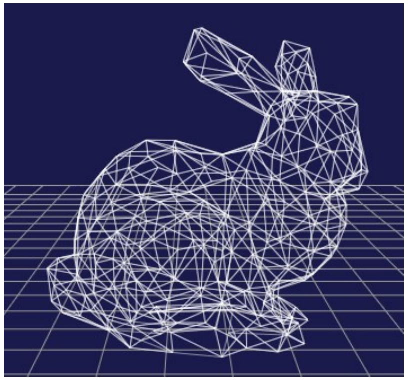


- Rendering time $\Theta(n + a)$
- Not suitable for large scenes

Conclusion:

- We need output-sensitive algorithms
- Weak dependence of rendering time on scene complexity

Randomized z-Buffer



sample point selection

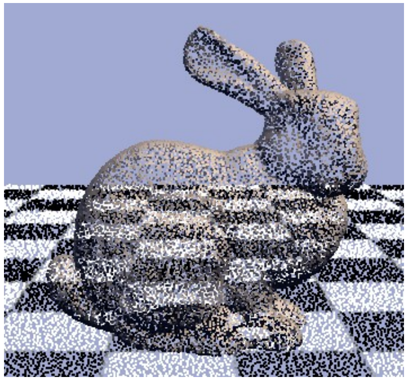
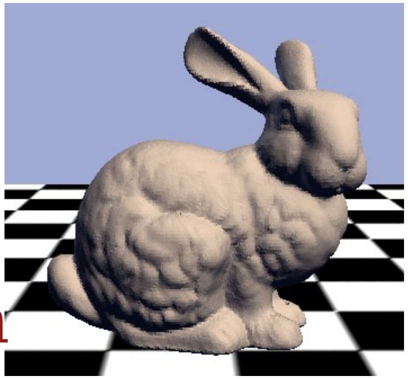


image recon-struction



triangles — sample points — bitmap

Outline of our algorithm:

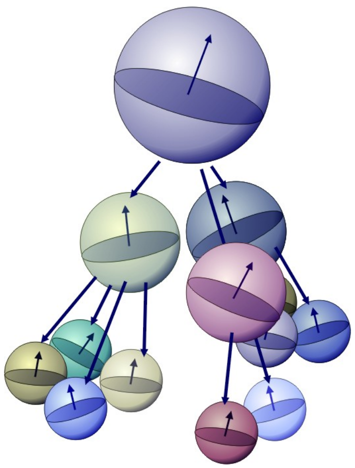
- Select sample points dynamically, approximately uniformly distributed on the projected areas of the objects
- Reconstruct an image out of the sample points

Running time: $O(a \cdot \log n)$

Related Techniques

Multi-resolution point sample rendering:

- QSplat [Rusinkiewicz, Levoy 2000]
- Surfels [Pfister et al. 2000]



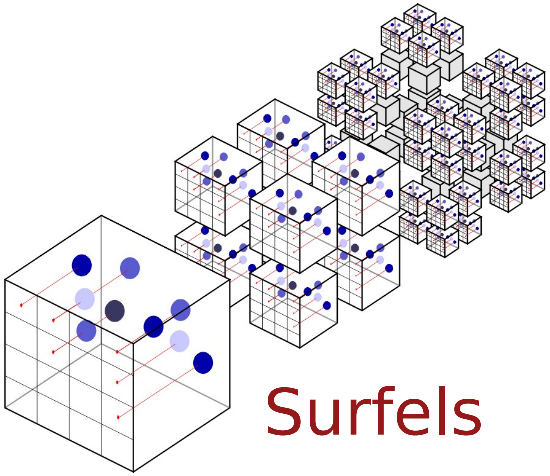
Qsplat

Approach:

- Precomputed hierarchy of point samples

Open problems:

- Fixed resolution
- Memory consumption
- Dynamic updates are expensive



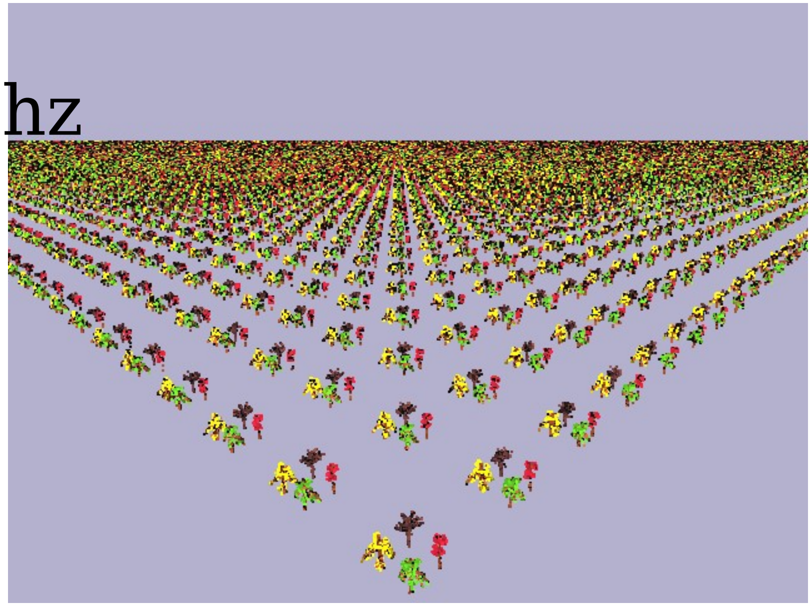
Surfels

Our Contribution

Randomized z-buffer:

- Fast on-the-fly generation of sample points
- Sampling time $O(a \cdot \log n)$ with $O(n)$ storage
- Efficient dynamic scene modifications
- Fallback to hardware z-buffer rendering for large triangles

Example: (800Mhz PC)



- 10^{14} triangles
- Sampling time: 4.3 sec
- Rendering time: 0.4 sec

Randomized z-Buffer: Image Reconstruction

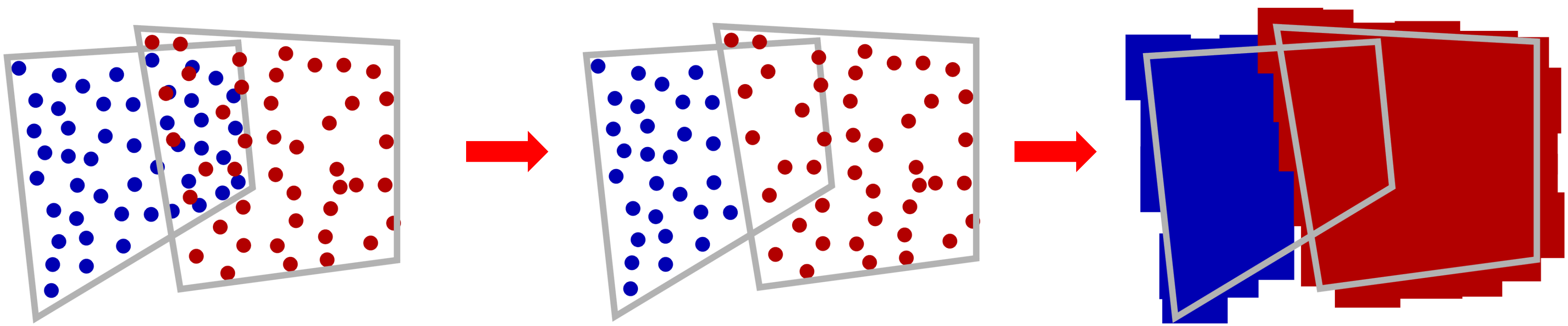
Image Reconstruction

Two problems:

Sample points

1. Reconstruction of occlusion

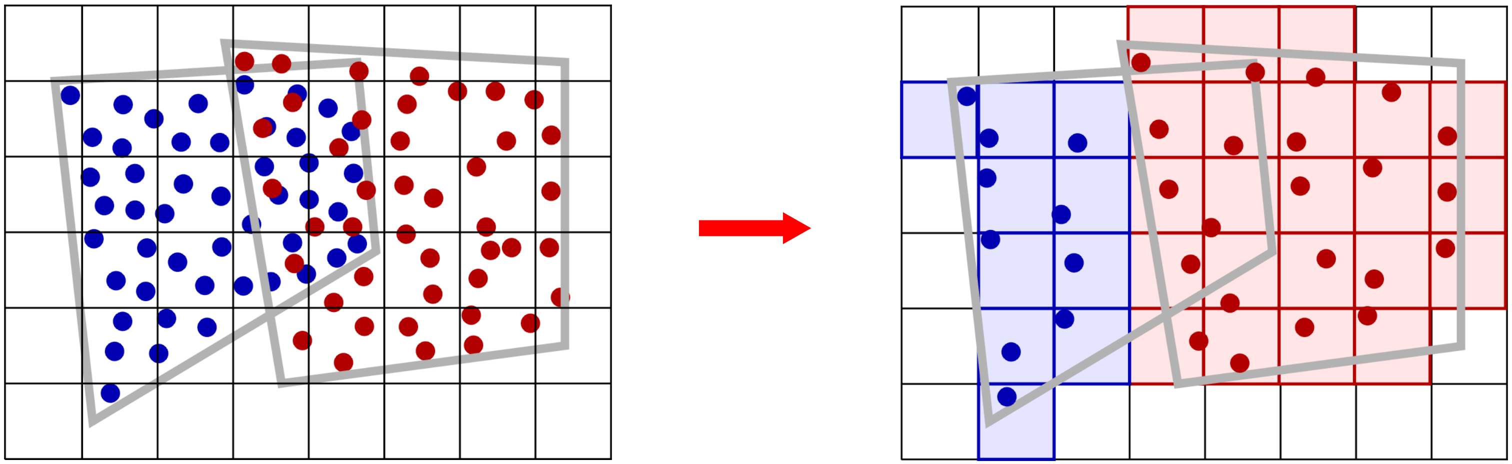
2. Filling



Remove adjacent points with larger depth

Scattered data interpolation

Per-Pixel Reconstruction



Per-pixel reconstruction:

Draw sample points into z-buffer

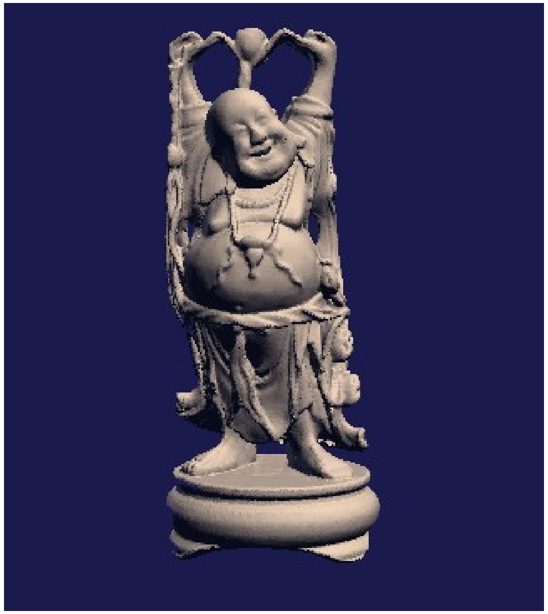
To cover all foreground area: $a \cdot \ln v$ sample points

a - Projected area (visible *and* occluded) [pixels]

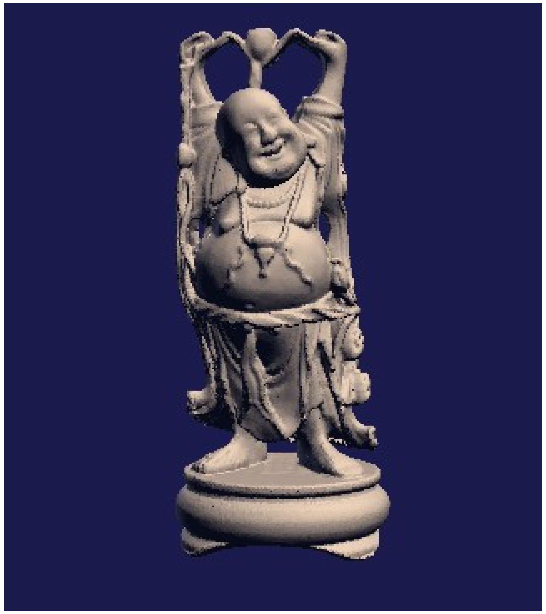
v - *Visible* projected area [pixels]

Splatting

Splatting: Draw colored splats of constant depth



$d = 1$
(110 msec)



$d = 2$
(30 msec)

$d = 5$
(7 msec)

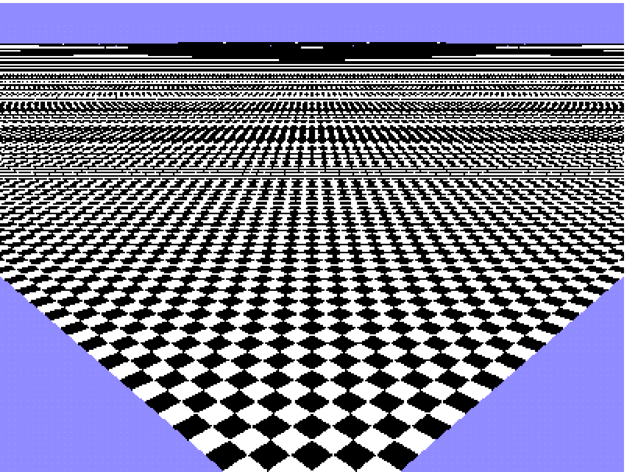
(d = splat size)

Gaussian Filtering

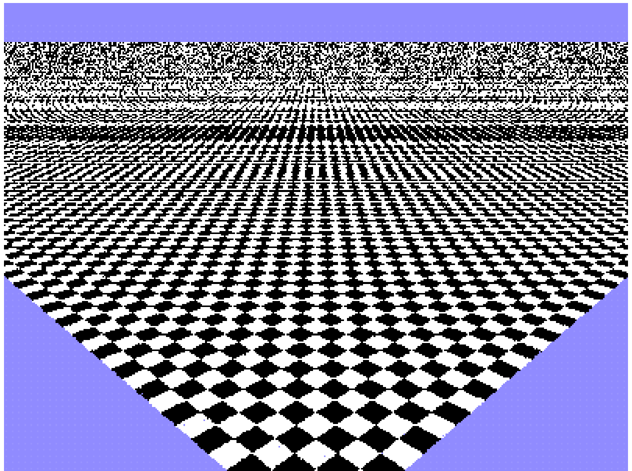
Gaussian Reconstruction:

- Use weighted averages in filling step
- Removes noise & aliasing
- Non-interactive reconstruction times (1-2 minutes)

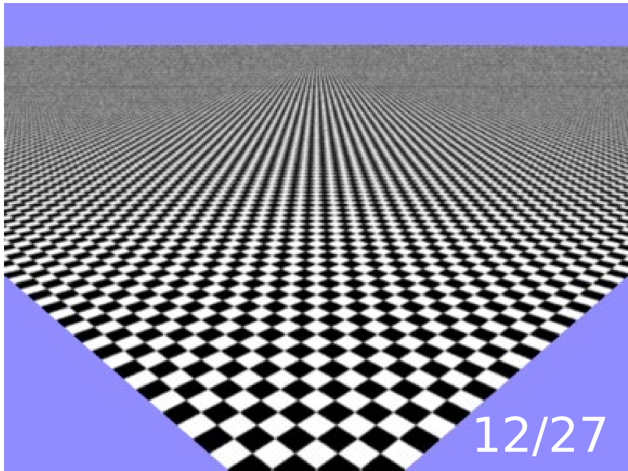
z-Buffer



Per-pixel reconstruction



Gaussian reconstruction



Choosing Sample Points

Projection Factor

Goal: Sample points uniformly distributed on the objects in the image plane

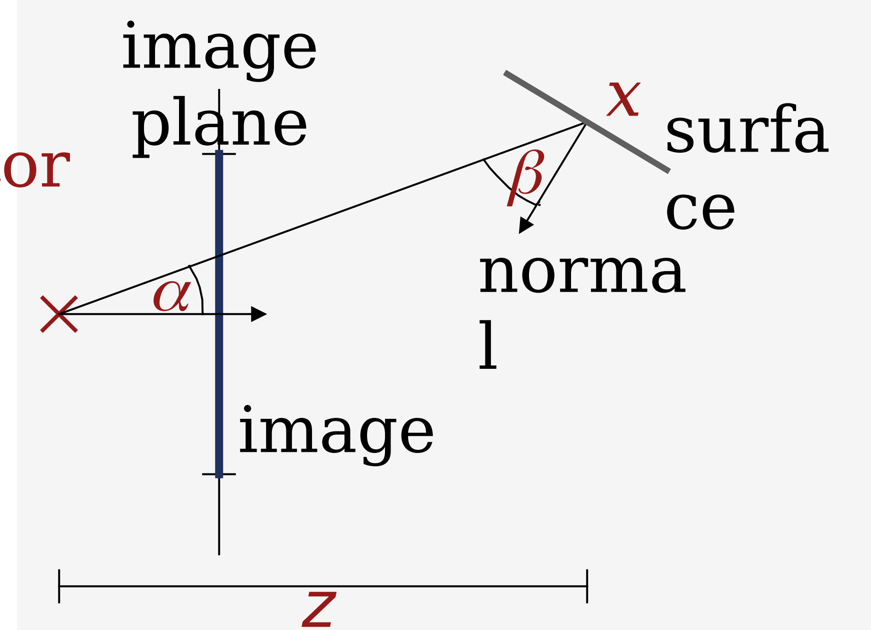
Projection factor: Factor by which an area fragment is scaled during perspective projection

depth factor

orientation factor

$$prj(x) = \frac{1}{z^2} \cdot \cos\beta \cdot \frac{1}{\cos\alpha}$$

distortion factor



Approximation (1)

Chose sample points: Projection factor as probability density in the view frustum

Efficient solution: Approximation algorithm

Idea: Approximation of the ideal distribution

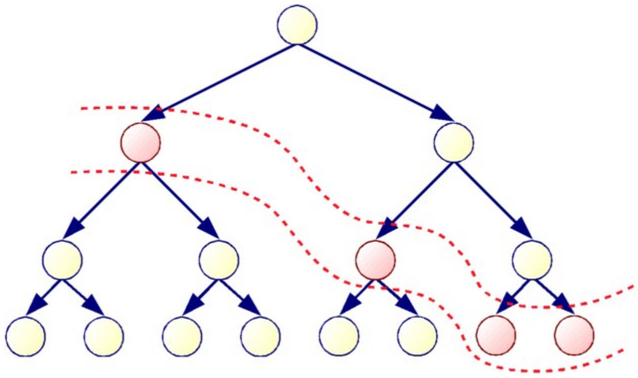
- Do not fall below minimum sampling density
- Exceeding the ideal sampling density leads to longer rendering time “only”

Approximation (2)

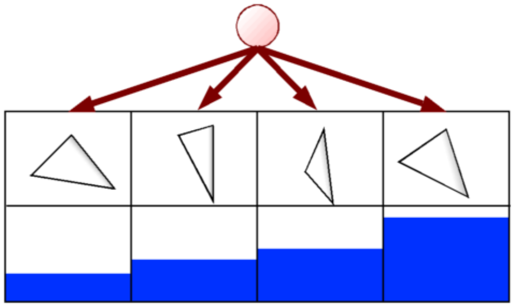
Approximation strategy:

- Precomputed hierarchical clustering of objects
- Online: choose groups of similar projection factor, calculate maximum projection factor
- In each group: distribution by unprojected area

Choosing Groups

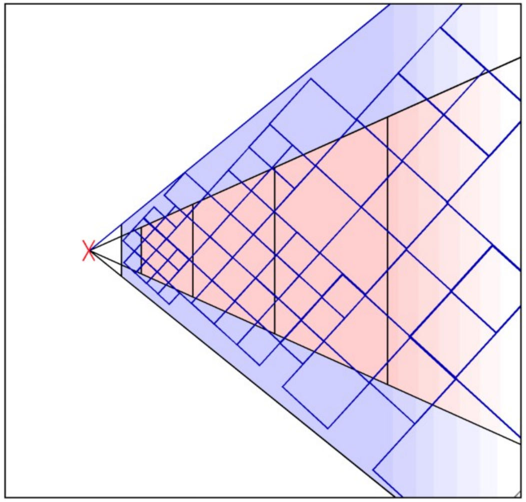


Choosing Triangles



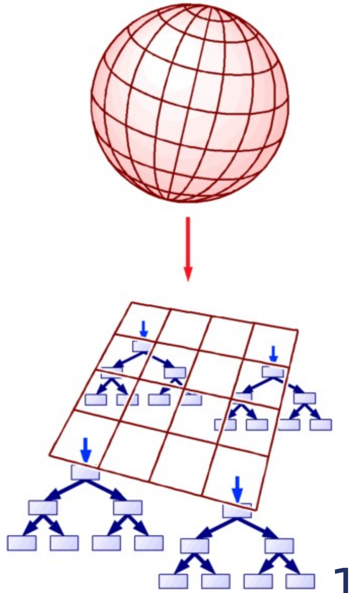
Grouping Objects

Spatial classification:



- Precomputed octree
- Choose boxes, in which $1/z^2$ does not vary by more than a constant
- $O(\log \tau)$ time, τ = minimal viewing distance / scene diameter

Classification by orientation:

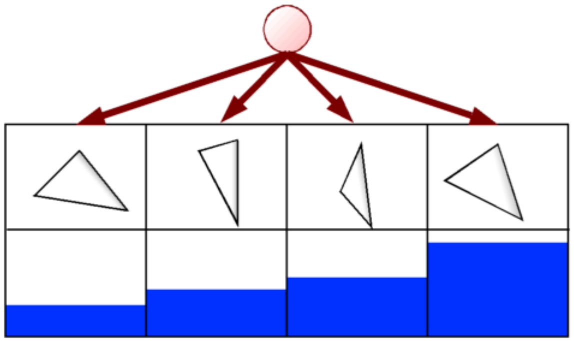


- Orientation classes
- Useful in special cases only

Analysis: neglect orientation factor

- Uniformly distributed surface normals
  $\Rightarrow$ overestimation factor = 4

Selection by Unprojected Area



Precomputation: Distribution List

- List of cumulated area values

Dynamic triangle selection:

- Chose random number uniformly from [0, *maxarea*]
- Binary search
- $O(\log n)$ running time for n triangles!

Sample point: Random linear combination

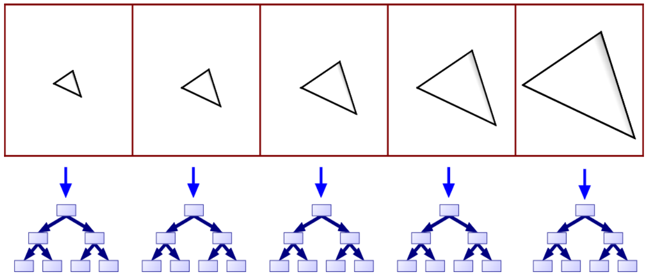
Improvements

Performance

Handling of large triangles:

- Projected area:
  triangle area $\times$ *projection factor*
- Classification by unprojected area
- Rasterize large triangles with z-buffer hardware

additional classification by triangle area:



Sample caching:

- Cache samples in spatial hierarchy nodes
- Speedup of up to factor 10
- Realtime performance on PC-hardware

Enhanced Data Structure

Dynamic modifications:

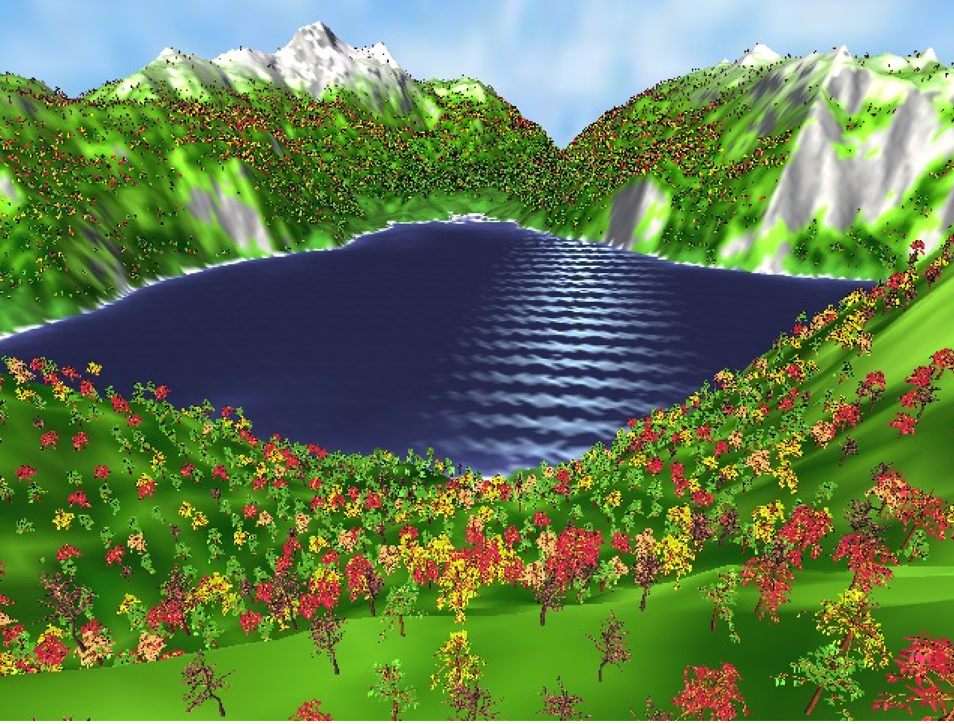
- Substitute dynamic search tree for distribution lists
- Insertion, deletion, modification in $O(h)$ (h = height of the spatial octree)

Efficient storage of highly complex scenes:

- Scene-graph based instantiation
- Storage $O(|SG|)$ instead of $O(n)$, $|SG|$ = size of scene graph

Example Renderings

Example: Landscape

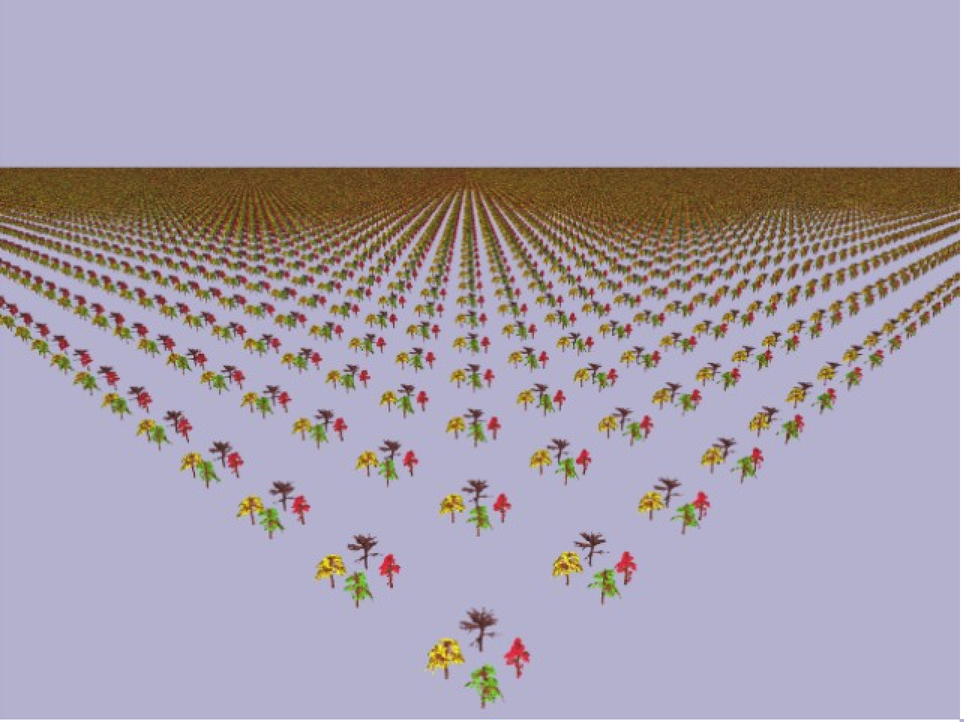


Phong lighting,
per pixel reconstruction,
rendering time: 19.2 sec

diffuse lighting, splatting (d=2),
sample caching,

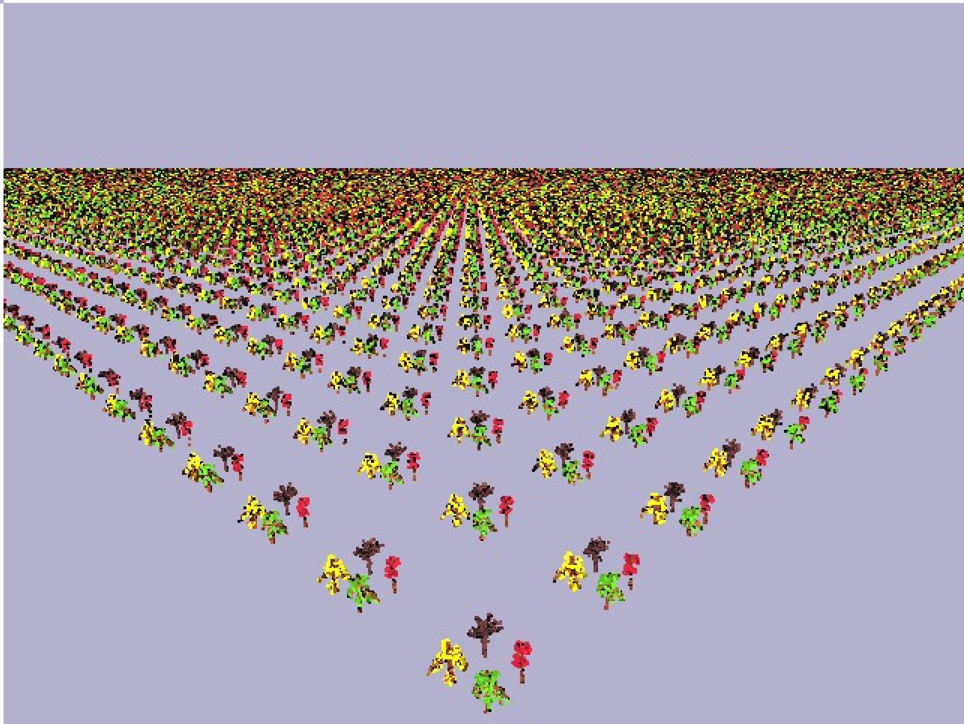
Complexity: 400 million triangles

Example: Forrest Scene



splatting, $d = 2$,
sample caching,
rendering time: 0.41 sec

Gaussian reconstruction,
rendering time: 120 sec



Complexity: 10^{14} triangles

Hardware: 800Mhz PC,

Future Work

Future Work

Future Directions:

- Modeling techniques for highly complex scenes
- Software framework
- Occlusion culling
- Global illumination